# Big Data Management Challenges, Approaches, Tools and their limitations

Michel Adiba, Juan-Carlos Castrejon-Castillo, Javier Alfonso Espinosa Oviedo, Genoveva Vargas-Solar, José-Luis Zechinelli-Martini

# Chapter 1. Big Data Management
## *Challenges, Approaches, Tools and their limitations*

Michel Adiba[15], Juan Carlos Castrejón[15], Javier A. Espinosa-Oviedo[13,]
Genoveva Vargas-Solar[134], José-Luis Zechinelli-Martini[2]
{ *michel.adiba, juan-carlos.castrejon, javier.espinosa, genoveva.vargas* }*@imag.fr*
*joseluis.zechinelli@udlap.mx*

Laboratory of Informatics of Grenoble (LIG) [1]
Fundación Universidad de las Américas, Puebla (UDLAP) [2]
Franco-Mexican Laboratory of Informatics and Automatic Control (LAFMIA) [3]
French Council of Scientific Research (CNRS) [4]
University of Grenoble (UdeG) [5]

**Abstract**

*Big Data* is the buzzword everyone talks about. Independently of the application domain, today there is a consensus about the V's characterizing Big Data: *Volume*, *Variety*, and *Velocity*. By focusing on Data Management issues and past experiences in the area of databases systems, this chapter examines the main challenges involved in the three V's of Big Data. Then it reviews the main characteristics of existing solutions for addressing each of the V's (e.g., NoSQL, parallel RDBMS, stream data management systems and complex event processing systems). Finally, it provides a classification of different functions offered by NewSQL systems and discusses their benefits and limitations for processing Big Data.

## 1. Introduction

Big Data is the buzzword everyone talks about since it concerns every human activity generating large quantities of digital data (e.g., science, government, economy). However, it is still difficult

to characterize the Big Data phenomenon since different points of view and disciplines attempt to address it. It is true that everyone sees behind the term a data deluge for processing and managing big volumes of bytes (Peta $10^{15}$, Exa $10^{18}$, Zetta $10^{21}$, Yotta $10^{24}$, etc.). But beyond this superficial vision, there is a consensus about the three V's [1] characterizing Big Data: *Volume*, *Variety* (different types of representations: structured, not-structured, graphs, etc.), and *Velocity* (streams of data produced continuously).

Big Data forces to view data mathematically (e.g., measures, values distribution) first and establish a context for it later. For instance, how can researchers use statistical tools and computer technologies to identify meaningful patterns of information? How shall significant data correlations be interpreted? What is the role of traditional forms of scientific theorizing and analytic models in assessing data? *What you really want to be doing is looking at the whole data set in ways that tell you things and answers questions that you're not asking* [2][3]. All these questions call for well-adapted infrastructures that can efficiently organize data, evaluate and optimize queries, and execute algorithms that require important computing and memory resources. With the evolution towards the cloud, data management requirements have to be revisited [4][5]. In such setting, it is possible to exploit parallelism for processing data, and thereby increasing availability and storage reliability thanks to replication. Organizing Big Data in persistence supports (cache, main memory or disk), dispatching processes, producing and delivering results implies having efficient and well-adapted data management infrastructures. These infrastructures are not completely available in existing systems. Therefore it is important to revisit and provide systems architectures that cope with Big Data characteristics. The key challenge is to hide the complexity for accessing and managing Big Data but also to provide interfaces for tuning them according to application requirements.

By focusing on Data Management issues, the chapter examines the main challenges involved in the three V's of Big Data and discusses systems architectures for proposing *V's model aware data management solutions*. Accordingly, the remainder of the chapter is organized as follows. Section 2 characterizes Big Data in terms of the V's model. In particular it insists on the aspects that lead to new challenges in data management and on expected characteristics of processed Big Data. Section 3 describes data processing platforms including parallel approaches, NoSQL systems, and Big Data management systems (BDMS). Section 3 introduces the life cycle of Big Data processing. It also describes possible application markets underlining the expected requirements that will lead to push the limits of what can be expected when fine-grained data are observed. Finally, section 5 concludes the chapter and discusses Big Data perspectives.

## 2. The Big Data V's

While some initial successes have already been achieved such as the Sloan Digital Sky Survey [6], genome databases, the library of Congress, etc., there remain many technical challenges that must be addressed to fully exploit Big Data potential. For instance, the *sheer size* of the data is a major challenge and is the one that is most easily recognized. However, there are challenges not just in *Volume*, but also in *Variety* (heterogeneity of data types, representation, and semantic interpretation) and *Velocity* (rate at which data arrive and the time in which it must be processed) [7].

### *2.1 Variety*

Data variety has been a recurrent issue since it has been possible to digitalize multimedia data and since the production of documents has become a day-by-day practice in organizations and in the domestic context. Continuous work has been done for modeling data and documents

that are digitalized in different formats. Raw data representation has been standardized (PDF documents, JPEG, GIF for images, MP3 for audio, etc.) and then coupled with data models in order to facilitate manipulation and information retrieval operations.

In the 1980's the relational model (structured data model) was defined on a solid theoretical basis namely mathematical relations and first order logic. The relational approach does an important distinction between schema (intention) and the extension of the relation. This dichotomy schema-data is fundamental for the database approach. Relations enable the manipulation of structured data independently of their physical representation in a computer. Given that a relation is a set of tuples that only contain atomic values, several consequences have to be considered. First a relation cannot contain repeated data; in a tuple an attribute cannot have as associated value a set, a table, or another relation; there cannot be an undefined or missing value in a tuple. These constraints led to extensions to the relational model, which is considered not expressive enough. The first approach was to relax the first normal form of relations and authorize attribute values to be of type relation. Generalizing the use of constructors of type Cartesian product and set, and then adding lists and tables led to the definition of the complex object model, which is more expressive. Attempts have been made to define Object-Oriented DBMS and these systems were characterized in [8].

The structured and semi-structured models (HTML, XML, JSON) are today managed by existing database management systems and by search engines exploring the Web and local files in computers. Semi-structured data are mostly electronic documents that emerged with the Web. We consider also that Object Oriented Databases influenced the JSON model (JavaScript Object Notation) today used as data exchange model on the Web. JSON is a generic format for textual

data derived from the Javascript objects notation.[1]

**Place Table 1.1 HERE**

Later, with the vague of NoSQL systems, other data models have emerged and are being used for dealing with Big Data. The key-value[2] data model associates a key to a simple or complex value. Records are distributed across nodes within a servers network using, for example, a hash function over the key. The key-value model being the simplest model is used for dealing with non-complex data like those in logs, user sessions, shopping cart data, that have to be retrieved fast and where there is no manipulation of the value elements. In the document model, data are semi-structured documents corresponding to nested structures that can be irregular (similar to markup languages like XML)[3]. In the column model data are grouped into columns in contrast to traditional relations stored in rows. Each element can have a different number of columns (non fixed schema)[4]. Document and column-family models are mainly used for event logging, blogging and Web analytics (counters on columns). The manipulation of documents is done on the content with few atomicity and isolation requirements and column-families are manipulated with concurrency and high throughput. Graph models provide concepts like nodes, edges, and navigation operations for representing respectively objects and querying operations. Nodes and edges can have properties of the form <`key, value`>[5]. Graph data models are adapted for highly connected data used for when information is retrieved based on

---

[1] Several NoSQL systems like CouchDB [44] proposed in 2005 and MongoDB in 2009 are based on JSON (see NoSQL section).
[2] Memcached, Redis and Riak are examples of systems that use this model.
[3] MongoDB and CouchDB are the most prominent examples of systems adopting this model.
[4] HBase, Cassandra and Hypertable are examples of systems that adopt this data model.
[5] Neo4J is an example of system that uses this model.

relationships. Every model has associated manipulation operations that are coupled to the data structure it relays on (see **Table 1.1**). The figure shows the way data can be looked up and retrieved: by key, aggregating data and navigating along relationships. For every possibility there are specific functions provided by the NoSQL systems API's.

Semantic content representations also appeared in order to support the Semantic Web (ontology languages like OWL, and tagging models like RDF) and lookup tools deployed on computers and other devices. For improving scalability, current research [9] is applying parallel models to the execution of reasoning engines where ontologies and linked data have millions of nodes and relationships.

This diversity is somehow the core of Big Data challenges (and of database integration in general), since it is no longer pertinent to expect to deal with standardized data formats, and to have generic models used for representing content. Rather than data models, the tendency is to have data representations that can encourage rapid manipulation, storage and retrieval of distributed heterogeneous, almost raw data. Key challenges are (i) to cope database construction (data cleaning) with short "time to market" (despite the volume of data and its production rate); (ii) to choose the right data model for a given data set considering data characteristics, the type of manipulation and processing operations applied to data, and "non functional properties" provided by the system. "Non functional properties" of systems include the performance of look up functions given the possibility of associating simple or complex indexing structures to data collections.

### *2.2 Volume*

The first thing anyone thinks about Big Data is its size [10][11]. In the era of Internet, social networks, mobile devices and sensors producing data continuously, the notion of size associated

to data collections has evolved very quickly [7][12]. Today it is normal for a person to produce and manage Terabytes of information in personal and mobile computing devices [13]. Managing large and rapidly increasing volumes of data has been a challenging issue for many decades [14][13][15]. In the past, this challenge was mitigated by processors getting faster, following Moore's law, to provide us with the resources needed to cope with increasing volumes of data. But, here is a fundamental shift underway now: data volume is scaling faster than compute resources and CPU speeds are not significantly evolving. Cloud computing now aggregates multiple disparate workloads with varying performance goals (e.g., interactive services demand that the data processing engine return back an answer within a fixed response time cap) [5]. This level of resources sharing on expensive and large clusters requires new ways of determining how to (i) run and execute data processing jobs to meet cost-effectively the goals of each workload; and (ii) deal with system failures that occur more frequently on large clusters [12][11].

### *2.3 Velocity*

The term 'velocity' refers to the speed of data generation and the time for processing it. Big Data are the result of a fine-grained continuous reading of the environment, society, and organizations, natural and social phenomena. Observations are done in different conditions and with different devices and therefore Big Data are raw heterogeneous continuously produced data (i.e., streams) that must be processed for extracting useful information. A stream is a sequence (a priori infinite) of couples $(t_i, v_i)$ where $t_i$ is a time stamp and $v_i$ is a (simple or complex) value. There are two possible interpretations for a stream and either as a *data flow* or as an *event flow*. Two aspects must be considered: data and users mobility in a spatio-temporal context, and the existence of several data or event streams produced continuously that must be processed under real time constraints more or less strict.

Several works have addressed stream processing and have proposed Stream Management Systems (SMS) and Complex Event Processing (CEP). Briefly, the difference between both approaches relies on the semantics associated to the data. For instance, a SMS compute the average of temperature during the day, while a CEP test whether the temperature in a room is not higher than a threshold and if it is then react in consequence[16].

There are two main reasons to consider stream processing. First it is not possible to store the data in their entirety: in order to keep storage requirements practical some level of analysis must occur as the data streams in. The issue is not just the velocity of the incoming data: it is possible to stream fast-moving data into bulk storage for later batch processing, for example. The importance lies in the speed of the feedback loop, taking data from input to decision. The second reason to consider streaming is where the application mandates immediate response to the data. Thanks to the rise of mobile applications and online gaming this is an increasingly common situation.

## 3. Big Data Processing Platforms

Schematically, (relational) databases and Web data management have evolved in parallel. On one hand, there is the success of relational, parallel Database Management Systems (DBMS) with SQL. On the other hand, specific distributed systems for managing Web data were developed using the Map-Reduce model (see NoSQL systems). The classical database approach seemed non suitable to manage Web data because it required successive design phases (ETL: Extract Transform, Load). These phases were considered too rigid with respect to (i) non-structured data sources (e.g. documents); (ii) specific Web applications; (iii) high availability and scalability for an increasing number of users. Furthermore, DBMS are expensive systems with few open source proposals like MySQL-C. Big Data processing platforms offer strategies with

good performance despite the data volume and greedy processing algorithms. Today these strategies start to converge and exploit the best of both worlds. From an application point of view, the question is how they can be combined to exploit Big Data?

Three architectures have emerged to address Big Data analytics: NoSQL systems, Map-Reduce/Hadoop, and extended RDBMS. At the beginning, these architectures were implemented as completely separate systems. Today, the trend is to develop innovative hybrid combinations of the three architectures. For example, Cisco and Oracle have delivered the first-ever enterprise class NoSQL solution deployed on Cloudera for harnessing large volumes of real-time unstructured and semi structured Big Data [17][18].

### *3.1 NoSQL Systems*

At the end of the 1990's with the development of the Web and companies like Google, Yahoo!, Facebook or Amazon, offering efficient data management solutions became crucial. Monolithic SQL databases built for OLTP and OLAP were rejected as being too expensive, too complex, and/or not fast enough. The « Not Only SQL »[6] movement was born [19][20]. For instance, Google and Amazon developed their own answers (BigTable and Dynamo, respectively) to meet these needs, and then, the Apache open-source community created corresponding clones like HBase and Cassandra, two of today's most popular and scalable key-value stores. Several systems have emerged providing their own implementations of the different data models and specific manipulation functions. The site of NoSQL databases[7] provides a list of existing systems.

The main objectives of NoSQL systems are to increase scalability and extensibility, using classic

---

[7] The site http://nosql-database.org/ gathers 150 systems with different models and functions.

hardware. They also ensure reliability, fault tolerance, and good performance despite the increasing number of requests. The common characteristic of these systems is that they enable data manipulation without having to define a schema, so data are mainly semi-structured. Thereby, they avoid schema definition and database loading after data cleaning. The characteristics of these systems architecture are: horizontal extensibility for executing distributed simple operations on a multitude of servers; data partitioning and duplication on several servers (data sharding); low level interfaces for accessing the systems; concurrent access model more flexible than ACID transactions; distributed indexing and in memory storage; easy data structure evolution [5]. They provide basic Create Read Update Delete (CRUD) functions adapted for efficiently manipulating data structures according to their underlying data model.

### *3.2 Parallel Data Processing with Map-Reduce*

Map-Reduce is a programming model (MR [21]) for developing data processing by defining the functions Map and Reduce inspired by functional programming. It is possible to parallelize data processing by partitioning and duplicating data on N machines, assuming that data are managed by a distributed file system. The function Map implements an operation based on a divide and conquer strategy. The idea is to divide a data set modeled as `<key, value>` tuples into smaller data sets that can be processed independently. Both key and value elements can be of atomic or complex data types. Reduce combines the result of the Map function and reduces them by applying aggregation or summation functions. Developers must write these functions but they can rely on a platform that provides tools for coordinating the jobs executing these functions in parallel.

MR execution platforms consist, in general, of a distributed file system for storing input and output data and a process manager that coordinates parallel jobs execution. The first proposal,

namely the GFS-Google File System [22] appeared in 2003. The architecture consists of millions of interconnected share nothing machines for incrementing data availability and reducing response time; ensuring the reliability of the system duplicating both servers and data; exploiting the inherent parallelism of the architecture. This system processes URL, user data (most recent query results), and geographical information (location of the points of interest in a region, like boutiques, hotels, streets satellite images). The scale factor is high with millions of URL, different versions of pages, millions of users, millions of queries per second, hundreds of TB for satellite images. GFS offers a familiar byte-stream-based file view of data randomly partitioned over hundreds or even thousands of nodes in a cluster [23]. GFS is coupled with the programming model MR, to enable programmers to process Big Data by writing two user-defined functions, Map and Reduce. Google builds distributed applications with MR functions that are then executed by Hadoop [24] an open source MR execution platform.

Concerning the architecture, the Hadoop nodes are of different types and all have HDFS (Highly Distributed File System open source version of GFS [24]) and MR layers. HDFS is based on a master slave architecture consisting of a master, the *name node* (an instance per cluster), that manages meta-data for the data in each node; the *backup node* of the name node; the *data node*, an instance deployed in every machine of the cluster, manages data storage on disk. Similarly, MR components are distributed and manage jobs: with the *name node* there is a *job tracker*, and with the *data node*, a *task tracker* executing the MR functions on local data. This architecture scales towards millions of nodes that can manage PB of data. Data blocks are large enough for avoiding overcharging data access from disk. Data are stored in each block in three copies for ensuring fault tolerance, instead of using mirror disks (e.g. RAID). Data location is invisible to the upper layers and this opens room for optimizations like those implemented by

RDBMS.

Several works have been devoted for implementing relational operators using MR [25][26][27]. For example, a selection corresponds to a filter for Map. Projection is also expressed by a simple Map function. For a join, Map [27][28][29] provides tuples with the join attribute as key and Reduce performs the join. Thus a relational query can be implemented as a set of parallel MR jobs. This kind of approach is provided by Hive [30] or Pig [31] built on top of Hadoop that provide SQL like languages for programming data processing and associated execution environments.

### *3.3 Big Data Management Systems*

Parallel RDBMS use a "shared-nothing" architecture [32] where data are spread over a cluster based on a partitioning strategy, usually hash-based, but sometimes range or random partitioning. Queries are processed by using parallel, hash-based divide-and-conquer techniques [33]. Parallel architectures address two factors expecting to get a linear behavior: (1) linear speed up, using the double of resources so that the program runs twice fast, for example processing 10TB using 8 nodes instead of 4 and thereby dividing the execution time by two; (2) linear speedup, double the resources used for processing a database twice bigger in the same response time, for example process 10TB on 4 nodes and 4 disks and then process 20TB using 8 nodes and 8 disks. In reality, linear speedup is a theoretical vision since other factors are to be considered: interconnection time between processors, load balancing among distributed servers.

[34] compares RDBMS with the Hadoop platforms for specific data analytics tasks. RDBMS seem to achieve good performances under the condition of appropriate tuning as good as Hadoop solutions or even better in some cases. These systems are robust and stable, but expensive (no open source system available). In contrast, Hadoop platforms are accessible and

require less tuning efforts but they lack transparency and require algorithmic design effort when implementing binary operations like joins. Application development is always ad-hoc in Hadoop since there is no general one fits all solution like in RDBMS. Thus, [34] believes that there is a need for designing a highly scalable platform for next-generation information storage, search, and analytics: a Big Data Management System (or BDMS). This can be done by combining and extending ideas drawn from semi-structured data management, parallel databases, and first-generation data intensive computing platforms (notably Hadoop/ HDFS). ASTERIX[8] aims to be able to access, ingest, store, index, query, analyze, and publish very large quantities of semi-structured data. The design of the ASTERIX BDMS is well-suited to handling use cases that range all the way from rigid, relation-like data collections—whose structures are well understood and largely invariant—to flexible and more complex data, where little is planned ahead of time and the data instances are highly variant and self-describing.

### *3.4 Discussion*

Different comparisons have been made between RDBMS, NoSQL and Hadoop platforms [35][36][37]. NoSQL systems seem to provide too simple indexing strategies in comparison to relational DBMS. They encourage the programming of queries using the MR model in contrast to the declarative and optimized approach of relational DBMS. In contrast, Hadoop can process data directly without defining a schema as for relational DBMS. Hadoop and NoSQL solutions do not provide general "one fits all" systems, and it requires a lot of expertise and programming effort for implementing solutions. In all cases, extensibility and fault tolerance are aspects addressed by all systems according to their characteristics. In response to Hadoop and NoSQL

---

[8] http://asterix.ics.uci.edu

systems drawbacks, recently, Google Cloud Dataflow[9] has been proposed. The objective is to make data and analytics accessible to everyone [38] through a data-centric model. With Dataflow programmers can easily express data processing pipeline, monitor its execution, and get actionable insights from data, without having to deploy clusters, tuning configuration parameters, and optimizing resource usage.

## 4. Big Data Life Cycle and Applications

"*Big Data is a paradigm shift in how we think about data assets, where do we collect them, how do we analyze them, and how do we monetize the insights from the analysis*," says Kimball [39]. Therefore, it is important to analyze Big Data life cycle and underline the challenges in each of these phases. Only analytic results matter. In the case of Big Data the whole processing phases are challenging: what to keep and what to throw? In which conditions are data acquired and cleaned? What is volatile or persistent, for how long? Are results transient or should they be used in further analysis?

### *4.1 Data Acquisition*

Data is collected from some data-generating source. Much data are of no interest, and they can be filtered and compressed by orders of magnitude [12][10][40][11]. One challenge is to define these filters in such a way that they do not discard useful information. Data acquisition calls for research in the information reduction science that can intelligently process raw data to a size that its users can handle while not missing the needle in the haystack. Furthermore, "on-line" analysis techniques are required to process streaming data on the fly, since it is not possible to store first and reduce afterwards.

---

[9] http://googlecloudplatform.blogspot.fr/2014/06/sneak-peek-google-cloud-dataflow-a-cloud-native-data-processing-service.html

Furthermore, it is necessary to automatically generate the right metadata to describe what data are recorded and measured. Another important issue is data provenance. Recording information about the data at its birth is not useful unless this information can be interpreted and carried along through the data analysis pipeline [41]. Thus research is required for both generating suitable metadata and designing data systems that carry the provenance of data and its metadata through data analysis pipelines.

### *4.2 Data Cleaning*

Given the heterogeneity of data flood, it is not enough merely to record it and store it into a repository. This requires differences in data structure and semantics to be expressed in computer understandable forms, and then "robotically" tractable. There is a strong body of work in data integration that can provide some of the answers. However, considerable additional work is required to achieve automated error-free difference resolution. Usually, there are many different ways to store the same information, each of them having their advantages and drawbacks. We must enable other professionals, such as domain scientists, to create effective database designs, either through devising tools to assist them in the design process or through forgoing the design process completely and developing techniques so that databases can be used effectively in the absence of intelligent database design.

### *4.3 Data Analysis and Mining*

Methods for querying and mining Big Data are fundamentally different from traditional statistical analysis on small samples. Big Data are often noisy, dynamic, heterogeneous, inter-related and untrustworthy. Nevertheless, even noisy Big Data could be more valuable than tiny samples. Indeed, general statistics obtained from frequent patterns and correlation analysis usually overpower individual fluctuations and often disclose more reliable hidden patterns and

knowledge.

Big Data are enabling the next generation of interactive data analysis with real-time answers. In the future, queries towards Big Data will be automatically generated for content creation on websites, to populate hot-lists or recommendations, and to provide an ad hoc analysis of data sets to decide whether to keep or to discard them [42]. Scaling complex query processing techniques to terabytes while enabling interactive response times is a major open research problem today.

Analytical pipelines can often involve multiple steps, with built in assumptions. By studying how best to capture, store, and query provenance, it is possible to create an infrastructure to interpret analytical results and to repeat the analysis with different assumptions, parameters, or data sets. Frequently, it is data visualization that allows Big Data to unleash its true impact. Visualization can help to produce and comprehend insights from Big Data. Visual.ly, Tableau, Vizify, D3.js, R, are simple and powerful tools for quickly discovering new things in increasingly large datasets.

### *4.4 Big Data Aware Applications*

Today, organizations and researchers see tremendous potential value and insight to be gained by warehousing the emerging wealth of digital information [43]: (i) increase the effectiveness of their marketing and customer service efforts; (ii) sentiment analysis; (iii) track the progress of epidemics; (iv) studying tweets and social networks to understand how information of various kinds spreads and/or how it can be more effectively utilized for the public good. In a broad range of application areas, data are being collected at unprecedented scale. Decisions that previously were based on guesswork, or on painstakingly constructed models of reality, can now be made based on the data itself.

The Sloan Digital Sky Survey [6] has become a central resource for astronomers worldwide. The field of astronomy was transformed. In old days, taking pictures of the sky was a large part of an astronomer's job. Today these pictures are all in a database already and the astronomer's task is to find interesting objects and phenomena. In the biological sciences, there is now a well-established tradition of depositing scientific data into a public repository, and also of creating public databases for use by other scientists. In fact, there is an entire discipline of bioinformatics that is largely devoted to the analysis of such data. As technology advances, particularly with the advent of Next Generation Sequencing, the size and number of experimental data sets available is increasing exponentially.

In the context of education, imagine a world in which we have access to a huge database where every detailed measure of every student's academic performance is collected. These data could be used to design the most effective approaches to education, starting from reading, writing, and math, to advanced, college-level, and courses. We are far from having access to such data, but there are powerful trends in this direction. In particular, there is a strong trend for massive Web deployment of educational activities, and this will generate an increasingly large amount of detailed data about students' performance.

Companies are able to quantify aspects of human behavior that were not accessible before. Social networks, news stream, and smart grid, are a way of measuring "conversation", "interest", and "activity". Machine-learning algorithms and Big Data tools can identify whom to follow (e.g. in social networks) to understand how events and news stories resonate, and even to find dates [9].

## 5. Conclusions and Perspectives

The Big Data wave has several impacts on existing approaches for managing data. First,

data are deployed on distributed and parallel architectures. Second, Big Data affects the type and accuracy of the models that can be derived. Big Data implies collecting, cleaning, storing and analyzing information streams. Each processing phase calls for greedy algorithms, statistics, models, that must scale and be performed efficiently. Scaling data management depends on the type of applications using data analytics results: critical tasks require on-line data processing while more analytic tasks may accept longer production time. Applying different algorithms produces results of different precision, accuracy, etc. (i.e., veracity).

These complex processes call for new efficient data management techniques that can scale and be adapted to the traditional data processing chain: storage, memory management (caching), filtering, cleaning. New storage technologies, for instance, do not have the same large spread in performance between the sequential and random I/O performance. This requires a rethinking of how to design storage systems and every aspect of data processing, including query processing algorithms, query scheduling, database design, concurrency control methods and recovery methods [33]. It is also important to keep track of the type of processes applied to data and the conditions in which they were performed, since the processes must be reproduced, for instance for scientific applications. These techniques must be guided by hardware characteristics, for example memory, storage and computing resources and the way they are consumed. Particularly, in cloud architectures that are guided by “pay as you go” business models.

Future Big Data management systems should take into account the analytic requirements of the applications, the characteristics of data (production rate, formats, how critical they are, size, validity interval), the resources required and the economic cost for providing useful analytic models that can better support decision making, recommendation, knowledge discovery, and data science tasks.

**References**


[1] D. Laney, "3D Data Management: Controlling Data Volume, Velocity & Variety," META-Group, 2001.

[2] B. Grinter, "A big data confession," Interactions, vol. 20, no. 4, Jul. 2013.

[3] A. Halevy, P. Norvig, and F. Pereira, "The Unreasonable Effectiveness of Data," IEEE Intell. Syst., vol. 24, no. 2, Mar. 2009.

[4] S. Chaudhuri, "What next?: a half-dozen data management research goals for big data and the cloud," in Proc. of the 31st PODS Symposium on Principles of Database Systems (PODS'12), 2012.

[5] S. Mohammad, S. Breß, and E. Schallehn, "Cloud Data Management : A Short Overview and Comparison of Current Approaches," in 24th GI-Workshop on Foundations of Databases, 2012.

[6] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. VandenBerg, "The SDSS skyserver: public access to the sloan digital sky server data," in Proc. of the Int. Conf. on Management of data (SIGMOD'02), 2002.

[7] H. A.K and Madam Prabhu D., "No problem with Big Data. What do you mean by Big?," Journal-of-Informatics, pp. 30–32, 2012.

[8] M. Atkinson, D. Dewitt, D. Maier, K. Dittrich, and S. Zdonik, "The Object-Oriented Database System Manifesto," in Building an object-oriented database system, 1992, pp. 1–17.

[9] J. Langford, "Parallel machine learning on big data," XRDS Crossroads, ACM Mag. Students, vol. 19, no. 1, Sep. 2012.

[10] P. Lyman, H. R. Varian, J. Dunn, A. Strygin, and K. Swearingen, "How Much Information?," in Counting-the-Numbers, 2000, vol. 6, no. 2.

[11] P. C. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch, and G. Lapis, Understanding Big Data. McGraw-Hill, 2011.

[12] R. Apps and R. Scale, Big Data Sourcebook. 2014.

[13] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou, "The Researcher's Guide to the Data Deluge : Querying a Scientific Database in Just a Few Seconds," Proc. VLDB Endow., vol. 4, no. 12, 2011.

[14] L. Hoffmann, "Looking back at big data," Commun. ACM, vol. 56, no. 4, Apr. 2013.

[15] A. Kleiner, M. Jordan, T. Ameet, and S. Purnamrita, “The Big Data Bootstrap,” in Proc. of the 29th Int. Conference on Machine Learning, 2012.

[16] H. Andrade, B. Gedik, and D. Turaga, Fundamentals of Stream Processing. Cambridge University Press, 2014.

[17] Oracle Corporation, “Oracle NoSQL Database Compared to MongoDB,” White-Paper, 2011.

[18] P. Zikopoulos, D. DeRoos, K. Parasuraman, T. Deutsch, J. Giles, and D. Corrigan, Harness the Power of Big Data. McGraw-Hill, 2013.

[19] R. Cattell, “Scalable SQL and NoSQL data stores,” SIGMOD Rec., vol. 39, no. 4, May 2011.

[20] P. J. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot. Addison Wesley, 2012.

[21] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, Jan. 2008.

[22] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” in Proc. of the 19th ACM SOSP Symposium on Operating Systems Principles (SOSP’03), 2003, vol. 37, no. 5.

[23] M. Cafarella, A. Halevy, W. Hsieh, S. Muthukrishnan, R. Bayardo, O. Benjelloun, V. Ganapathy, Y. Matias, R. Pike, and R. Srikant, “Data Management Projects at Google,” SIGMOD Rec., vol. 37, no. 1, 2008.

[24] D. Borthakur, “HDFS Architecture Guide,” Apache-Report, pp. 1–13, 2010.

[25] J. Dittrich and J.-A. Quiané-Ruiz, “Efficient big data processing in Hadoop MapReduce,” Proc. VLDB Endow., vol. 5, no. 12, Aug. 2012.

[26] F. Li, B. C. Ooi, M. T. Özsu, and S. Wu, “Distributed data management using MapReduce,” ACM Comput. Surv., vol. 46, no. 3, Feb. 2014.

[27] A. Okcan and M. Riedewald, “Processing theta-joins using MapReduce,” in Proc. of the 2011 ACM SIGMOD Int. Conference on Management of Data (SIGMOD ’11), 2011.

[28] J. D. Ullman, “Designing good MapReduce algorithms,” XRDS Crossroads, ACM Mag. Students, vol. 19, no. 1, Sep. 2012.

[29] J. Chandar, “Join Algorithms using Map / Reduce,” Slides, 2010.

[30] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop," in Proc. of the 26th ICDE Int. Conference on Data Engineering (ICDE'10), 2010.

[31] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in Proc. of the Int. Conf. on Management of data (SIGMOD'08), 2008.

[32] P. Valduriez, "Parallel Techniques for Big Data Outline of the Talk," Slides, 2013.

[33] V. R. Borkar, M. J. Carey, and C. Li, "Big data platforms: What's next?," XRDS Crossroads, ACM Mag. Students, vol. 19, no. 1, Sep. 2012.

[34] M. Stonebraker, D. Abadi, and D. DeWitt, "MapReduce and parallel DBMSs: friends or foes?," Communications-of-the-ACM, 2010.

[35] 451-Research, "Mysql vs. nosql and newsql: 2011-2015," Report, 2012.

[36] C. Mohan, "History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla," in Proc. of the 16th EDBT Int. Conference on Extending Database Technology (EDBT'13), 2013.

[37] V. Borkar, M. J. Carey, and C. Li, "Inside ' Big Data Management ': Ogres , Onions , or Parfaits ?," 2012.

[38] K. Ren, Y. Kwon, M. Balazinska, and B. Howe, "Hadoop's adolescence: an analysis of Hadoop usage in scientific workloads," Proc. VLDB Endow., vol. 6, no. 10, Aug. 2013.

[39] Chris Forsyth, "For Big Data Analytics There' s No Such Thing as Too Big," Forsyth-Communications, 2012.

[40] Chris Sherman, "What's the Big Deal About Big Data ?," Online Search., vol. 38, no. 2, 2013.

[41] P. Agrawal, O. Benjelloun, A. Das Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom, "Trio: A System for Data, Uncertainty, and Lineage," in Proc. of the 32nd Int. Conf. on Very Large Databases (VLDB'06), 2006.

[42] S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki, "Here are my Data Files. Here are my Queries. Where are my Results?," in Proc. of the 5th CIDR Biennial Conference on Innovative Data Systems Research (CIDR'11), 2011.

[43] K. Michael and K. Miller, "Big Data: New opportunities and new challenges," Computer (Long. Beach. Calif)., vol. 46, no. 6, 2013.

[44] CouchBase, "NoSQL Database Technology," Report, 2014.

## Table Captions

Table 1.1 Data models proposed by NoSQL systems

| Type | Operation | Data Models | Description |
|---|---|---|---|
| **Key** | Read(*key*) | Key-Value, Graph, Document, Column-Family, Relational | *Read* a single record based on *primary key* |
| | Insert(*key*, *value*) | | *Insert* a single record based on *primary key* |
| | Update(*key*, *value*) | | *Update* a single record based on *primary key* |
| | Delete(*key*) | | *Delete* a single record based on *primary key* |
| **Aggregation** | Read(*pattern*) | Document, Column-Family, Relational | *Read* all entities that conform to the specified *pattern* |
| | Insert(*value*) | | *Insert* a single record with the specified *pattern* |
| | Update(*pattern*, *value*) | | *Update* all entities that conform to the specified *pattern* |
| | Delete(*pattern*) | | *Delete* all entities that conform to the specified *pattern* |
| **Connection** | Read(*relationship*) | Graph, Relational | *Read* all entities that conform to the specified *relationship* |
| | Insert(*relationship*) | | *Create* a *relationship* between entities |
| | Update(*oldRel*, *newRel*) | | *Update* a *relationship* between entities |
| | Delete(*relationship*) | | *Delete* a *relationship* between entities |